DOE Workshop for Industry Software Developers

*March 31, 2011*

# HPCToolkit • DynInst

## *Open Source Tools From CScADS*

`hpctoolkit.org` `dyninst.org`


**John Mellor-Crummey**

RICE

**Barton Miller**

THE UNIVERSITY of WISCONSIN MADISON

**Nathan Tallent**

RICE


Center for Scalable Application Development Software

`cscads.rice.edu`

# Why Tools? "Why doesn't my app work?"

- Complex architectures are hard to program
  - multicore/manycore multi-socket nodes
    - hardware threading, out-of-order pipeline, SIMD units
  - multi-level memory hierarchies
    - non-uniform latency memory and cache accesses
  - heterogeneity: SIMD, GPUs
  - massive scale: 200K cores
- Complex software is hard to monitor and analyze
  - parallelism: threads, locks, synchronization, nested parallelism
  - modularity: object-orientation, generics, meta-programming
  - layers: composing libraries, multilingual apps, coupled apps
  - compilers: transform code; may not achieve high performance
- How do I find bottlenecks? How do I find errors?
  - tools must not only work, but deliver actionable insight
    - insight that justifies specific actions

# The Role of Performance Tools

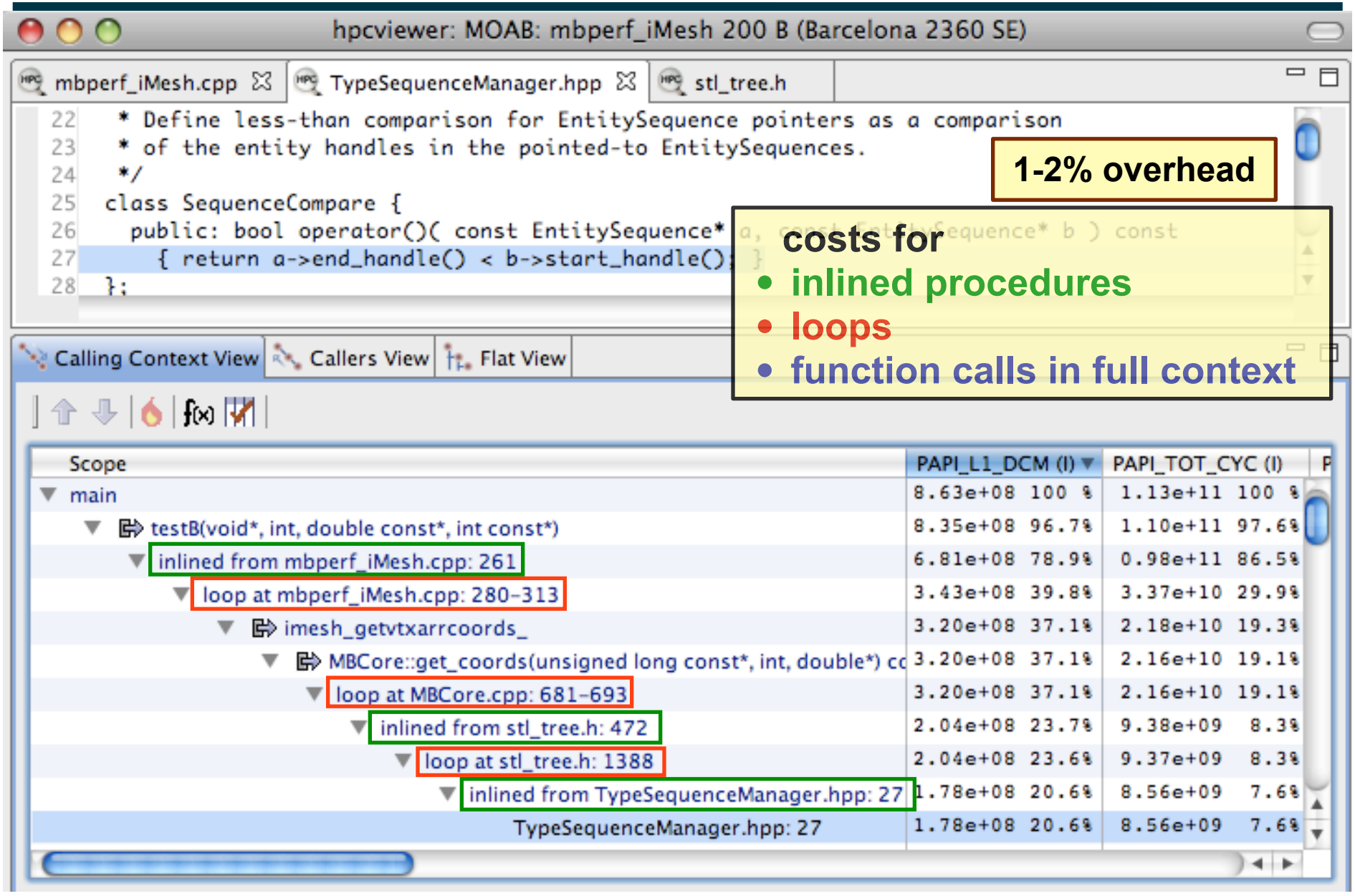
**Pinpoint and diagnose bottlenecks in parallel codes**

- Are there parallel scaling bottlenecks at any level?
- Are applications making the most of node performance?
- What are the rate limiting factors for an application?
  - mismatch between application needs and system capabilities?
    - memory bandwidth or latency?
    - lock contention?
    - communication bandwidth or latency?
- What is the expected benefit of fixing bottlenecks?
- What type and level of effort is necessary?
  - tune existing implementation
  - overhaul implementation to better match architecture capabilities
  - new algorithms

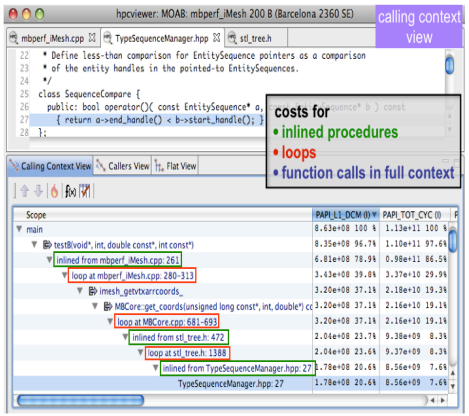
# Performance Tools Challenges

- Deliver actionable insight
  - pinpoint and explain problems in terms of source code
  - pinpoint ~~hot spots~~ inefficiency
- Combine accurate & precise measurement
  - attribute metrics to the statement, loop and data-object level
  - avoid high overheads
  - avoid systematic measurement error
- Measure configuration of interest, not a mock up
  - large multi-lingual, fully optimized, parallel, production code
  - binary-only libraries, partially stripped
  - dynamic loading or static binaries
- Scale within and across nodes
  - multithreaded codes
  - large-scale parallelism (200K cores)

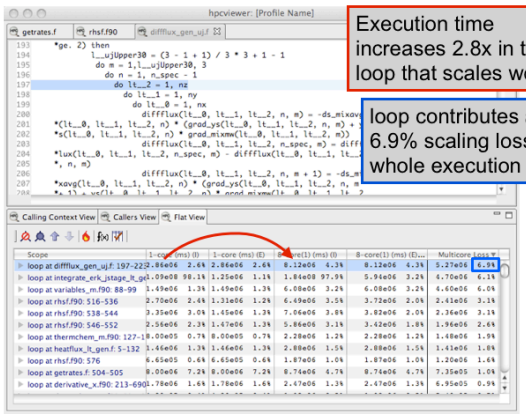
# HPCToolkit: Detailed Performance Metrics



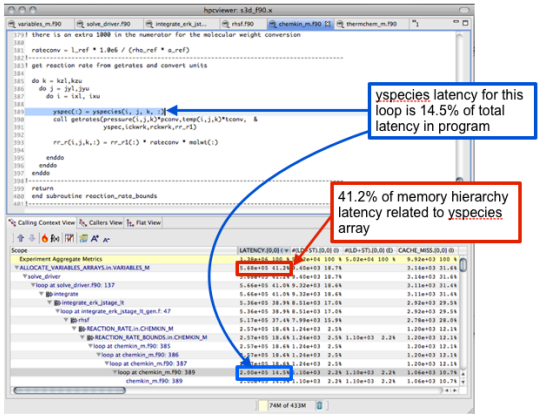
# HPCToolkit Capabilities at a Glance



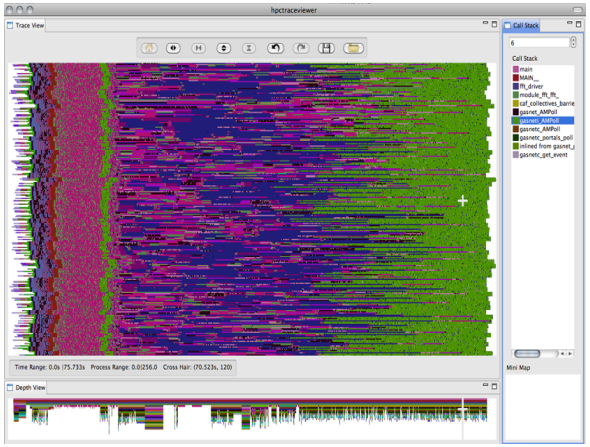

Attribute Costs to Code



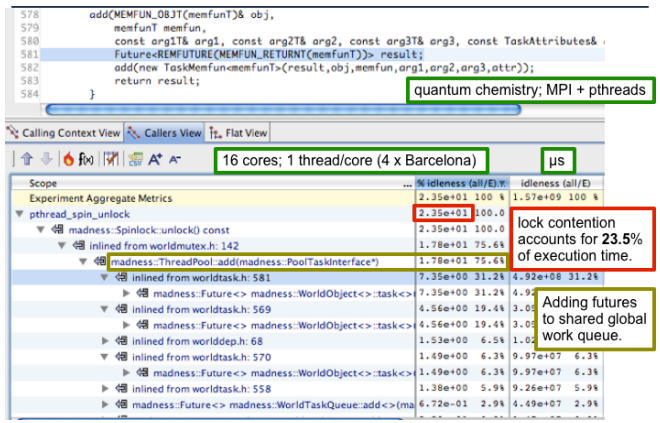

Pinpoint & Quantify Scaling Bottlenecks



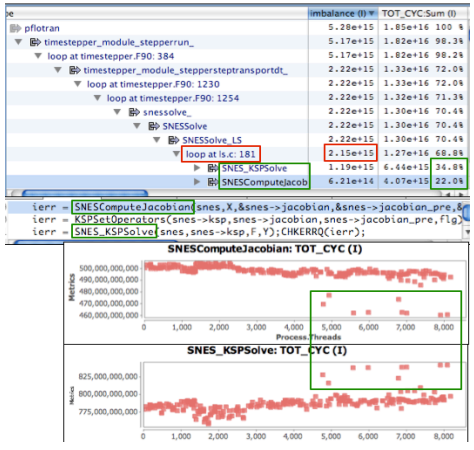

Associate Costs with Data



Analyze Behavior over Time




Shift Blame from Symptoms to Causes




Assess Imbalance and Variability

# "Performance Tools for Leadership Computing" - John Mellor-Crummey, Rice University

## ASCR SciDAC-2 Computer Science Highlight

### Objectives

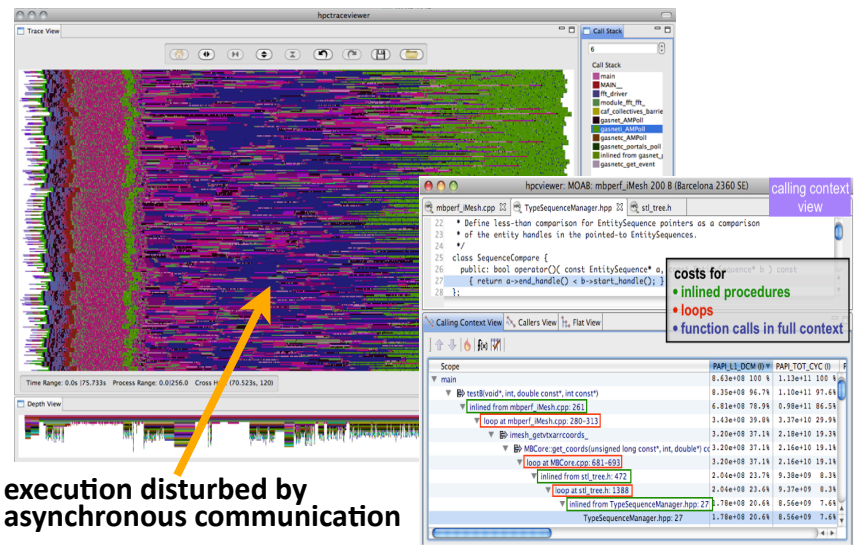
- Accurately measure performance of parallel codes on leadership computing systems
- Pinpoint and quantify losses both within and across nodes in scalable parallel programs
- Provide actionable insight by tying losses and opportunities to both code and data

### Impact

Used HPCToolkit to help DOE science teams understand and fix code performance problems

- Lock contention in MADNESS
- Data locality losses in S3D
- AMR scaling bottlenecks in FLASH
- Load imbalance in PFlotran simulations

**User interfaces tie costs to code and display behavior over time**




execution disturbed by asynchronous communication

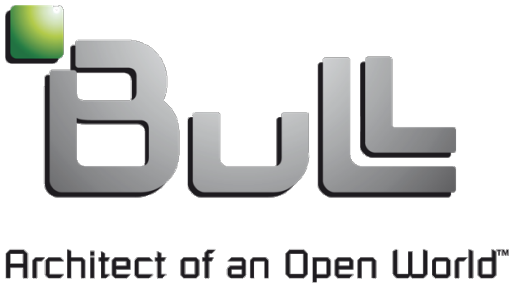
### Accomplishments

- Precise measurement and attribution using on-the-fly binary analysis - *Best paper PLDI09*
- Novel techniques to pinpoint & quantify (CCPE10)
  - scalability losses within & across nodes (ICS07,SC09)
  - inefficient multithreading (PPoPP09, IEEE Comp. 09)
  - lock contention (PPoPP10)
  - load imbalance (SC10)
  - data centric issues (CGO11)
- Insight into transient behavior with sampling (ICS11)
- HPCToolkit released as supported product on Bull and SciCortex systems; under evaluation by IBM

U.S. DEPARTMENT OF ENERGY | Office of Science

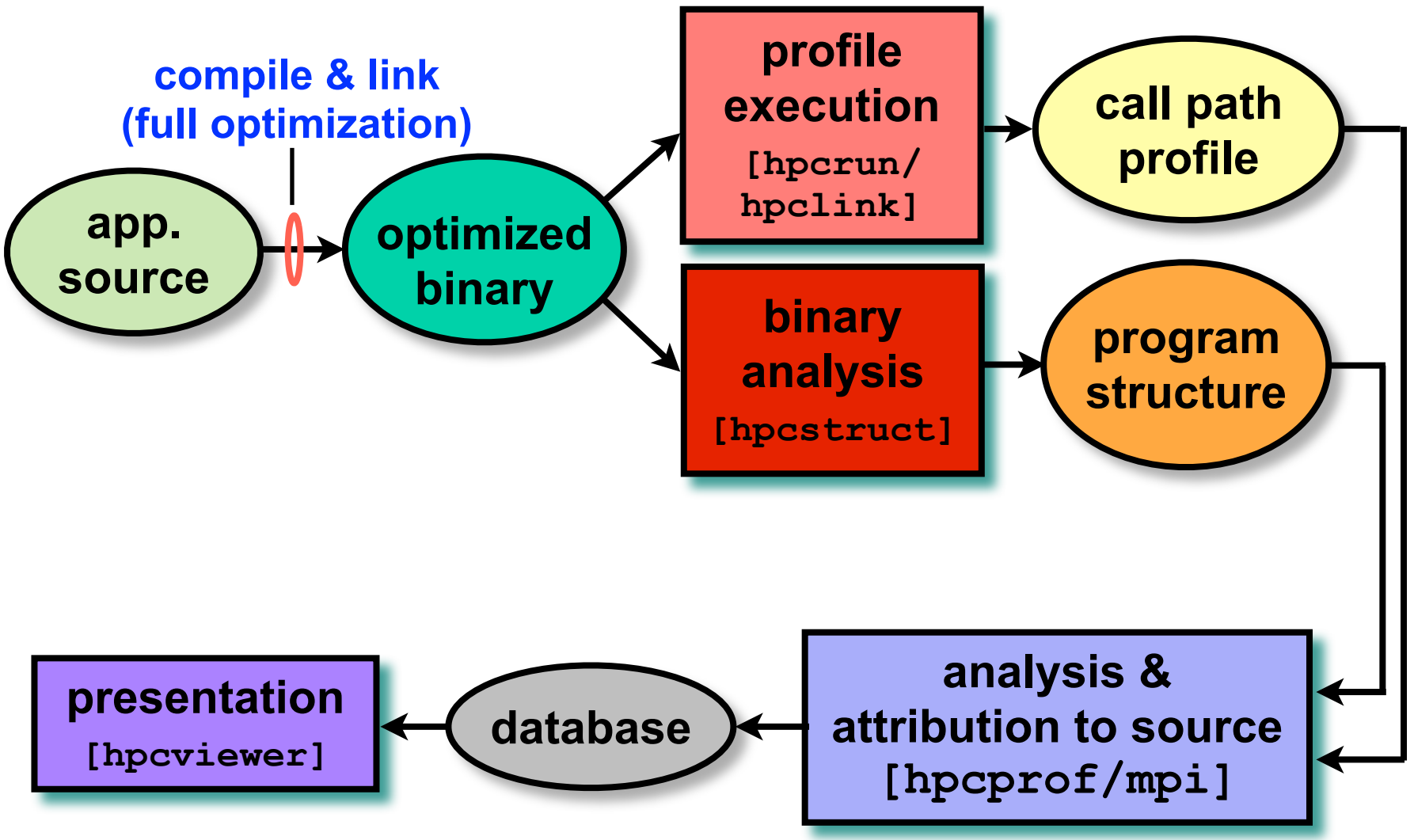
# HPCToolkit and Industry

- **Bull**
  - ships HPCToolkit with systems
- **SiCortex (low power clusters)**
  - shipped HPCToolkit with systems
- **Samara Technology Group**
  - packages HPCToolkit as a key tool used by its consultants
- **Total R&D (energy)**
  - used HPCToolkit extensively in performance studies of seismic code
- **WesternGeco (energy)**
  - studied performance of seismic code
- **IBM**
  - evaluating for POWER7 systems






# HPCToolkit at a Glance



**Dyninst:** Analyze, instrument, and control binary programs:

- Instrumentation: add tracing, checking, sandboxing, monitoring, etc. to your already-compiled code

  *Binary rewriting:* patch an a.out/.exe/.so/.dll to add or modify code.

  *Dynamic instrumentation:* add or remove instrumentation code from a *running* program.

- Analysis: Binary code parsing, Control- and data-flow analysis, symbol table access, instruction disassembly.
- Program Control: Create/attach to a process, start, stop, event detection, read, and write process address space.

# Guiding Principles

- Work on program binaries:
  - don't need source code, even for proprietary libraries.
  - functions even if code is stripped (no symbol tables)
- Clean Abstractions
  - Hide complexity
  - Necessary for portability
  - Quickly build new tools
- Portability
  - Same interface across multiple systems
  - System differences not visible

# The Toolkits

**DyninstAPI:** overall package for binary program analysis, instrumentation, and control (x86 32 and 64 bit, Power 32 and 64 bit, IBM BlueGene, Cray XT)

**SymtabAPI:** read, understand, and update program symbol tables (PE/PDB, ELF/DWARF)

**StackwalkerAPI:** runtime walk of process stacks, handling optimized code frame and exceptions.

**ProcControlAPI:** create/attach to process, monitor state change events, control process, and read/write address space (Linux, Windows, Cray XT, IBM BG)

# The Toolkits

**InstructionAPI:** parse a machine instruction into fields, providing a machine-independent representation.

**ParseAPI:** control- and data-flow analysis of the binary program, even in the face of stripped code.

**DataflowAPI:** data dependence analysis of the binary code, including forward and backwards slicing.

**DynC API:** generate program instrumentation code sequences from C code snippets.

- **DynInst components are shipped by Cray (in ATP)**
- **DynInst has been shipped by IBM (in DPCL)**
- **SymTabAPI used by HPCToolkit**

# CScADS Tools Technology Highlights

- **HPCToolkit (BSD License)** `hpctoolkit.org`
  - hpcrun call path profiler
  - libmonitor library for process/thread monitoring
  - hpcstruct binary analysis to recover program structure
  - hpcprof analysis tool
  - hpcviewer call path profile presentation tool
  - hpctraceviewer call path trace presentation tool
- **DynInst (LGPL License)** `dyninst.org`
  - dynamic binary instrumentation
  - process control
  - symbol table parsing and generation
  - instruction encoding and decoding
  - binary control-flow and data-flow analysis
  - first and third party stack unwinding

`cscads.rice.edu`